Documentation

Index

1 XML structure

The description of workflows is based on the following XML elements:

- Sequence
  A Sequence is a collection of workflow elements.
- Flow
  A Flow is a collection of Sequences. If proper compensation is made when creating the following element structure, this can be changed.
- Activity
  An activity is a single workflow entity

```
<sequence name="seq1" owner="">
    <activity name="act1" owner="Mad Max" />
    <activity name="act2" owner="Toad" />
    <activity name="act3" owner="Mr. Bean" />
    <flow name="flow1" owner="">
        <sequence name="seq2" owner="">
            <activity name="act4" owner="Allen" />
            <activity name="act5" owner="Miss Piggy" />
        </sequence>
        <sequence name="seq3" owner="">
            <activity name="act6" owner="John Doe" />
        </sequence>
    </flow>
    <activity name="act7" owner="GB" />
</sequence>
```

2 Reading from XML

Reading from XML is performed by the *PerformReadXML* method, witch locates the XML file. The file path is hard coded at this point, but handles English and Danish platforms.
ReadXML is called using the path as parameter, and the XML file is read using the *XmlTextReader* class.
The XML elements found in this way is added to a *ListedElements* object, which inherits an IEnumerable interface, so that it may be used through a ‘foreach’.
This initial read can be removed, since it in practice represents a step, which can be performed as a part of the process of reading the XML and creating the object structure through the structure processing.
An *Element* contains information on Name, Type and Owner. These are used to demonstrate the purpose of the elements, and will probably be insufficient in a real implementation.

3 Processing

The processing of the object structure is performed in two steps, a step creating the element structure it self and a step processing the data (*size*) describing the visualization of the structure.

3.1 Structure processing

Processing of the elements in *ListedElements* is performed through the method *ProcessElements* of the *Processor* class.
This reads through the list of elements starting with index 0. The method switches on the Type of the found element, before processing the content of each element, creating a tree structure of elements.
This structure is used when drawing the flow chart.

3.2 Visualization (pre-) processing

The *CollectSize()* method runs through the nested structure mentioned above. This is done to update the elements herein before they can be drawn. The size of an element (e.g. a flow) depends on the size of its sub elements, which is why this has to be precalculated.

4 Drawing

When calling *Draw* on the first element of the *ListedElements* this element draws itself and calls *Draw* on its child-elements.

Drawing of an element is handled by adding a *UserControl* to the *MainWindow*. *VisualActivity* is such a user control.
A usercontrol is an empty control, which can be designed by the user (obviously). This means that a usercontrol can be customized to support the particular needs of a project. A user control is added to the *MainWindow* so that it can be shown on the screen.
The usage of these user controls supports automatic handling of click-events.

Drawing of the element structure is based on the heights and widths of the following illustration:

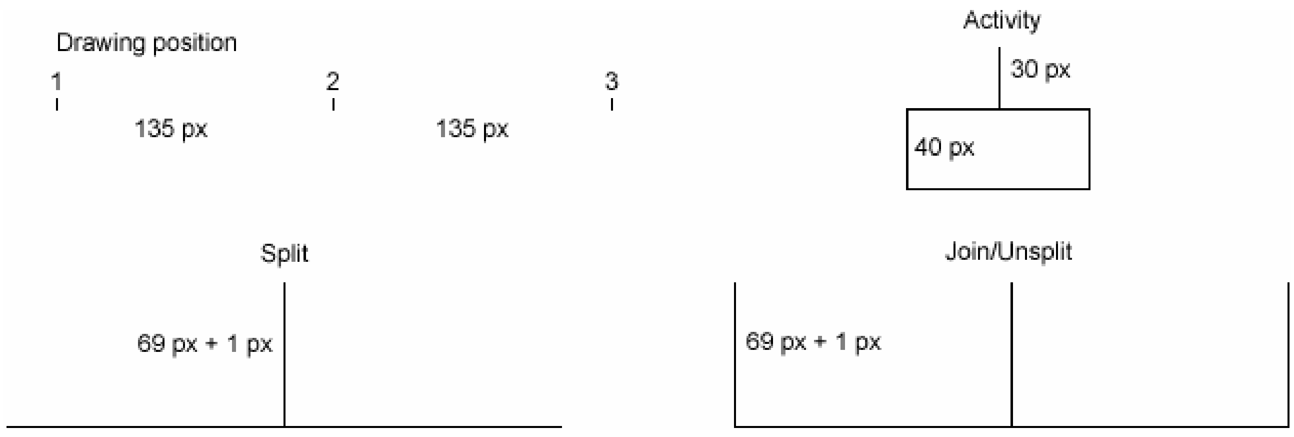


Figure 1: Heights and widths

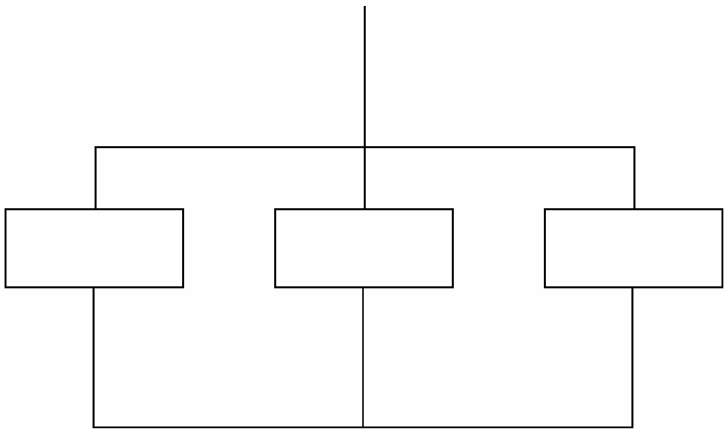
The Drawing positions are shown to illustrate the construction of these, and are not restricted to only three of such positions. When the drawing is performed the drawing positions are used to align the drawing of user controls, such that they can be drawn properly.

These sizes are defined from the size of the animated objects used in the implementation. They must be changed if the animated objects change, since the coordinates used for drawing is based on these sizes.

Collapsing of user controls is not implemented, though it is possible to add. The drawing method in a flow can for example be extended to skip drawing of its child elements. In this way the graph will be simplified, but not reduced in size. To do this the preprocessing has to be redone before the graph can be redrawn. The structure processing is omitted, since the element structure remains the same.

4.1 Specific drawing example

The following illustration shows a flow 3 parallel sequences (3 parallel flows)



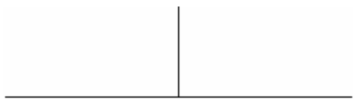
The drawing is performed by calculating the Drawing position (x-position) of the first child element of the flow (first, left part after the split). This is done using the equation:

$$Dp - \left(\frac{W_{\max}}{2}\right) + \frac{W_{child} - 1}{2}$$

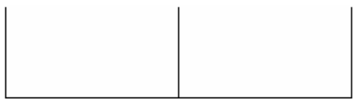
Where:
Dp = current drawpoint (depends on the position in the graph)
Wmax = the width of the sequence holding the flow and possible sibling flows
Wchild = the width of the child for which the drawing position is being calculated.

This can be found in the Draw method of the Flow.cs file.

The drawing of the split/header



and the join/unsplit/footer



is performed after the final child element (rightmost sequence) has been drawn. Then the coordinates of the children is known and the header and footer can be drawn and added to the *MainWindow*.